METHOD AND APPARATUS FOR EFFICIENT TIMEOUT MESSAGE MANAGEMENT

The present invention relates generally methods and apparatuses for managing messages in computer networks, and more particularly to a method and apparatus for managing TIMEOUT messages in a computer network, such as an IEEE 1394 bridge network for inter-coupling high performance serial buses.

The IEEE 1394 standard describes a high performance serial bus that has one of the most versatile interconnect technologies available. The IEEE 1394 high-speed serial bus is capable of transfer speeds of 100Mb/sec, 200Mb/sec, or even 400Mb/sec. These transfer speeds are available over twisted-pair wire, and the serial bus is hot-plug capable. Consequently, the IEEE 1394 serial bus can be used in many applications, including but not limited to, video streaming from a camcorder, controls for automobiles, and digital audio signals, such as MIDI signals.

In addition, the IEEE 1394 standard is an international standard for a low-cost digital interface that is used for integrating computing, communication, and entertainment into multimedia applications.

An important feature of the IEEE 1394 standard bus is its flexible topology that supports daisy chaining and branching for communication throughout a particular network. In an IEEE 1394 network, the serial bus architecture is defined in terms of nodes. A node is capable of being independently reset and is an identifiable and addressable entity. Each node is a logical entity having a unique address, which includes an identification ROM and multiple control registers. These control registers are a standardized set and can be reset independently of each other.

An IEEE 1394 network provides asynchronous transport that is a traditional memory-mapped, loaded and stored interface. During an asynchronous transport operation, a data request is sent to a specific address, and the entity having that address returns an acknowledgement.

In an IEEE 1394 network, there can be up to 1,023 logical buses and up to 63 nodes on each bus. If both the asynchronous packet sender and receiver are on the same bus, transaction timeout values can be óbtained according to IEEE draft standard 1394-1995.

However, if a sender exists on a different bus than a given receiver, the sender sends a TIMEOUT request message addressed to the bus to which the receiver is connected in order to obtain certain parameters (e.g., remote_timeout_seconds, remote_timeout_cycles, max_remote_payload and hop_count values) for a remote transaction between the two buses. The remote_timeout_seconds, the remote_timeout_cycles, the max_remote_payload and the hop_count values are sometimes referred to collectively as remote timeout values or parameters.

According to the draft standard for IEEE 1394.1 high performance serial bus bridges revision 1.00, each bridge on a path from a source bus to a destination bus shall intercept a TIMEOUT request message and update each field as follows: The TIMEOUT request message shall be forwarded by bridges on its route toward the destination bus. The last exit portal on the destination bus that intercepts the TIMEOUT request message also adds its local SPLIT_TIMEOUT value obtained according to IEEE 1394-1995 standard to the remote timeout field in the message and synthesizes a TIMEOUT response message that contains the result of the above calculation and sends it to the message sender.

According to IEEE1394.1 draft standard, when there is a node on a bus that sends a TIMEOUT request message to a destination bus, even if other nodes on the first source bus have already obtained the remote transaction timeout values for a remote transaction to the destination bus, a TIMEOUT request message shall be forwarded and processed by every bridge portal on the path from a source bus to the destination bus each and every time a TIMEOUT request message is initiated to the same destination bus.

Accordingly, the processing by each portal on a path between a source bus and a destination bus of subsequent TIMEOUT request messages from other nodes on the same source bus to the same destination bus is redundant, thereby straining bandwidth resources of a given network.

As another node on the source bus has already obtained the remote transaction timeout values of the destination bus, a node that needs the same remote transaction timeout values should be able to obtain these values from the node on the same bus that has previously received them. The redundancy of the transaction proceeding again to the same bridge portals on the path causes congestion by requiring unnecessary transaction times for another node that sends the TIMEOUT request message and has to wait for a corresponding response.

According to the draft standard IEEE P1394.1, Draft 1.0, a TIMEOUT message must be processed by each bridge along the way from the source bus to the destination bus to accumulate the timeout value each time the TIMEOUT request message is initiated.

The present invention is therefore directed to the problem of developing a method and apparatus for managing these TIMEOUT messages efficiently without loss of capability.

The present invention solves these and other problems by storing remote timeout information when a TIMEOUT message passes through a bridge for the first time, and using the stored information to synthesize a TIMEOUT response message for successive TIMEOUT request messages, thereby eliminating the need to forward TIMEOUT request messages further and reduce the message traffic and improve the efficiency of the network.

According to one aspect of the present invention, a method for communicating in a network having a plurality of nodes located on a plurality of buses coupled together by at least one bridge includes: checking by a portal upon receiving a timeout request message a flag to determine whether a set of stored remote timeout values is valid. If the stored set is valid, the portal synthesizes a timeout response message to the source bus using the set of stored remote timeout values.

According to another aspect of the present invention, if the stored set is not valid, the portal will take a variety of actions. In one exemplary embodiment, the portal will temporarily store the received set of remote timeout values; forward a new set of calculated remote timeout values in the forwarded timeout request message; intercept a timeout response message that corresponds to the forwarded timeout request message; store a revised set of remote timeout values based on a set of remote timeout response values included in intercepted timeout response message for later use with the valid flag updated; and forward a revised set of remote timeout response values as part of a timeout response message towards the source bus.

In a second exemplary embodiment, the portal will forward the timeout request message unchanged towards the destination bus, and another portal associated with the portal will upon receipt of a timeout response message update the set of remote timeout values and store them for later use based on a set of remote timeout response values included in the intercepted timeout response message, along with forwarding the updated set of remote timeout response values in a timeout response message towards the source bus.

FIG 1 illustrates an exemplary embodiment of a source bus and a destination bus linked serially by a series of bridge portals to which various aspects of the present invention are applicable.

FIG 2 depicts a detailed schematic of an exemplary embodiment of a register table for storing various flags and timeout values according to another aspect of the present invention.

FIG 3 depicts a flowchart of an exemplary embodiment of a method for intercepting a TIMEOUT response message and for storing remote timeout values according to yet another aspect of the present invention.

FIG 4 depicts a flowchart of another exemplary embodiment of a method for intercepting a TIMEOUT request message, and synthesizing and sending a TIMEOUT response message according to yet another aspect of the present invention.

FIG 5 depicts a flowchart of another exemplary embodiment of a method for efficiently communicating timeout messages in a bridged network.

FIG 6 depicts a flowchart of another exemplary embodiment of a method for efficiently communicating timeout messages in a bridged network.

It is worthy to note that any reference herein to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the invention. The appearances of the phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment.

In a related patent application by the same inventors, disclosed therein is a method and apparatus for providing an efficient TIMEOUT message management. According to the method disclosed therein, an entry portal on a source bus intercepts an initial TIMEOUT response message for a particular destination bus and stores its timeout value. For successive TIMEOUT request messages for the same destination bus, the entry portal on the source bus (or an intermediate bus) synthesizes the corresponding TIMEOUT response message using the stored remote timeout value without further forwarding of the TIMEOUT request message towards the destination bus, thus reducing the traffic on the network. By storing the TIMEOUT values at the entry portal on the source bus, however, only the bridge on the source bus can access this timeout value for future use.

The above described method for efficient TIMEOUT message management includes the following steps:

(a) implementing a register table in a portal, which table contains entries for storing respective remote timeout values from a local bus of a portal to a particular destination bus in a same net, in which an Nth entry of the register table corresponds to a bus ID of N;

(b) intercepting a TIMEOUT response message en route to a particular-addressed node by an exit portal if the TIMEOUT response message is addressed to the local bus of the portal, which TIMEOUT response message includes remote timeout values;

(c) storing the remote timeout values contained in the TIMEOUT response message intercepted in step (b) in a corresponding entry in the register table implemented in step (a);

(d) forwarding the TIMEOUT response message intercepted in step (b) to the particular-addressed node;

(e) intercepting by a portal of a TIMEOUT request message from an initial requester, if the remote timeout values from the local bus of the portal to the destination bus to which the intercepted TIMEOUT request message is addressed have been stored previously by step (c) in the register table recited in step (a);

(f) synthesizing by the portal of a corresponding TIMEOUT response message having the remote timeout values for a remote transaction from the local bus of the portal to the destination bus to which the intercepted TIMEOUT request from step (e) is addressed by one of the following steps:

(i) retrieving the remote timeout values from the register table if the initial requester of the TIMEOUT request message identified in step (e) is on the local bus of the portal; and

(ii) calculating the remote timeout values retrieved from the register table if the initial requester of the TIMEOUT request message identified in step (e) is not on the local bus of the portal, in which a max_remote_payload value is the smaller of max_remote payload values in either: (1) the intercepted TIMEOUT response message in step (b); or (2) the corresponding register table entry; and in which remote timeout seconds, remote timeout cycles and hop count values in the intercepted TIMEOUT request message are added to the corresponding register table entry to the destination bus, respectively; and

(g) sending the TIMEOUT response message synthesized in step (f) to the initial requester of the TIMEOUT request message intercepted in step (e).

With regard to step (a), in an exemplary embodiment, the register table includes at least 1023 entries. However, the number of entries may be different if the embodiment is employed in a serial bus other than the 1394, or according to other design requirements.

With regard to step (e), the portal will not forward the intercepted TIMEOUT request message to the destination bus, which is a different procedure than according to the IEEE P1394.1 draft standard.

With regard to step (f)-(ii), the remote_timeout_seconds, the remote_timeout_cycles and the hop_count values in the intercepted TIMEOUT request message are added to those in the table register entry corresponding to the destination bus ID, respectively. The max_remote_payload value is set to a smaller value between that in the intercepted TIMEOUT request message and that stored in the table register entry corresponding to the destination bus ID.

The register table recited in (a) may include a Random Access Memory (RAM) of a portal.

The source and destination buses may be connected in a serial path via one or more bus bridges. The bus may comprise part of a bridged network, which can be a 1394-bridged network.

Turning to FIG 1, shown therein is an example of an IEEE1394 serial bus net including a first or source bus 100, a second bus 110 and a third or destination bus 120; the first bus 100 and second bus 110 are connected by a bridge 130, and the second bus 110 and the third bus 120 are connected by a bridge 140.

Bridge 130 comprises portals 131 and 132, and bridge 140 comprises portals 141 and 142. Node 150 exists on the first bus 100 as a TIMEOUT request message sender, and node 160 exists on the third bus 120 as a destination of a TIMEOUT request message addressed node. A network configuration to which the invention applies should not be restricted only to this example though.

Firstly, the node 150 sends a TIMEOUT request message to the node 160 when obtaining the remote timeout values from the source bus 100 to the destination bus 120.

This TIMEOUT request message could be addressed to either node 140 or destination bus 120.

The TIMEOUT request message will be processed according to the IEEE1394.1 Bridge draft standard and forwarded to the last exit portal 142 on destination bus 120.

According to this exemplary method, upon reception of the TIMEOUT request message, the last exit portal 142 will calculate the remote timeout values according to the P1394.1 working draft, (the contents of which are herein incorporated by reference as background material as if repeated herein in their entirety) and will send a TIMEOUT response message with the calculated remote timeout information to the original TIMEOUT request message sender 150 on bus 100.

In addition, bus 110 also has node 155, which is shown for the following explanation. When node 155 sends a TIMEOUT request message toward bus 120, the TIMEOUT request message will be forwarded to portal 142 while the remote timeout values are calculated according to IEEE 1394.1 draft standard.

Subsequently, portal 142 sends a TIMEOUT response message with the remote timeout values to the original TIMEOUT message requester 155 on bus 110. Portal 141 intercepts the TIMEOUT response message sent by portal 142 since it is addressed to portal 141's local bus (i.e., bus 110) and stores the remote timeout values for a remote transaction from bus 110 to bus 120 in the TIMEOUT response message into the register table entry corresponding to the destination bus 120. Therefore, when node 150, 151, or portal 131 on bus 100 sends a TIMEOUT request message toward bus 120, the portal 141 shall intercept the TIMEOUT request message and synthesize its TIMEOUT response message with calculated remote timeout values according to one aspect of the present invention.

Portal 131 on source bus 100 will intercept the TIMEOUT response message, if it is addressed to its local bus 100, and store the remote timeout values found in the TIMEOUT response message into the entry of its internal register table 133 (shown in detail in FIG 2) corresponding to the bus ID of the destination bus 120.

The first entry portal forwards the TIMEOUT response message to the original requester on its local bus as explained in the IEEE 1394.1 bridge draft standard, the contents of which are herein incorporated by reference as background material as if repeated herein in its entirety.

The remote timeout values from the source bus 100 to the destination bus 120 that are stored in the register table 133 will be valid until either the source bus ID of the source bus 100, or the destination bus ID of the destination bus 120 becomes invalid or is cleared in terms of net update state noted in the P1394.1 working draft.

Thus, the above procedure can be used for intercepting a TIMEOUT response message and storing the timeout response message found in the message in the register table. Next, the procedure for intercepting a TIMEOUT request message and providing a synthesized TIMEOUT response message is explained below.

Subsequently, when portal 131 on bus 100 receives a TIMEOUT request message initiated by another node 151 (or possibly node 150) on the same source bus 100, the first entry portal 131 synthesizes a TIMEOUT response message containing the remote timeout values, which can be retrieved from an entry of its internal register table 133 corresponding to the destination bus 120, because the valid remote timeout values from the source bus 100 to the destination bus 120 have been stored into the register table by the entry portal 131 as explained above.

Subsequently, the first entry portal 131 on bus 100 sends this synthesized TIMEOUT response message to the TIMEOUT request message requester 151, instead of forwarding the TIMEOUT request message to the destination bus 120.

Thus, the first entry portal 131 on the source bus 100 can synthesize a TIMEOUT response message, if the register table of the entry portal 131 has stored the desired remote timeout values for a remote transaction from its local bus 100 to the destination bus 120 in the internal register table 133.

Turning to FIG 2, shown therein is an exemplary embodiment of a register table for storing timeout values for from its portal's local bus to each particular bus. While the register table may consist of 1023 entries for 1023 possible buses, the number of the table entries is not restricted to 1023 and could differ. Typically, the register table would include one or more entries for each possible bus.

Each table entry includes multiple fields. In this case, the fields shown are: a remote_timeout_seconds, a remote timeout cycles, a max_ remote_ payload and a hop_count, which are the same as those specified in a TIMEOUT message as defined in IEEE1394.1 bridge draft standard. Other fields may be included. FIG 2 shows at least one additional field reserved for another possible entry. Each timeout value as measured from its portal's local bus to a particular destination bus is stored in a same field of a corresponding table entry to the particular destination bus. This register table 133 could consist of RAM or any other type of storage known in the art.

In other words, in this embodiment, if a portal is a last exit portal, which forwards a TIMEOUT response message, addressed to the portal's local bus, and which contains remote timeout values for a transaction between its local bus and a particular remote destination bus, the TIMEOUT response message is processed according to the IEEE 1394.1 bridge draft standard, except that for the interception of the TIMEOUT response message and the storing of the remote timeout values in the TIMEOUT response message into the entry of the internal register table entry corresponding to the particular remote destination bus by previously disclosed steps (b) and (c).

In addition, subsequent to storage of the remote timeout values between the source bus 100 and the destination bus 120, the above embodiment method is thereafter utilized when other nodes on the same bus as the portal are connected to, such as the node 151 and/or possibly the same node 150, including bridge portal 131 on the same bus 100 that need remote time values from the local bus 100 to the same remote bus 120, by synthesizing a TIMEOUT response message containing the remote timeout values retrieved from the timeout values previously stored in the register table 133 of the first entry portal 131. Synthesis of the TIMEOUT response message and direct reply to the TIMEOUT request message sender significantly reduces congestion between the bus 100 and bus 120, and speeds obtainment of remote timeout values because the second requesting node 151 receives the remote timeout values from the local bus to the destination bus 120 much faster directly from the first entry portal 131 than it would by the conventional method according to IEEE1394.1 bridge draft standard.

The illustration of a network configuration shown in FIG 1 is for purposes of illustration only and not for limitation, and a person of ordinary skill in the art should understand that the network configuration is not restricted to the illustration, as any number of buses could be connected serially. When applying the present invention to the 1394 configuration, it is understood that the network configuration also can be varied so long as it is permissible according to the IEEE P1394.1 draft standard. For example, a source bus and/or a destination bus may be connected to more bridge portals and/or there may be more intermediate buses between the source bus and the destination bus.

The above embodiments also can be applied to a case in which portal 141 on intermediate bus 110 has already stored remote timeout values from the intermediate bus 110 to a particular destination bus 120 into the internal register table entry 143 corresponding to the destination bus 120 by step (b) and (c). The synthesized message may contain a smaller max_remote payload value of the intercepted message and the corresponding register table entry.

For example, node 155 on the bus 110 as a source bus for this transaction could send a TIMEOUT request message to the destination bus 120. According to the above embodiments, portal 141 on the source bus 110 intercepted the corresponding TIMEOUT response message sent by the portal 142 on the destination bus and stored the remote timeout values from the source bus 110 to the destination bus 120 into the internal register table entry 143 corresponding to the bus ID of the destination bus 120.

As described in step 425(ii) of FIG 4, when the portal 141 on the bus 110 receives a TIMEOUT request message issued by a node on the source bus 100 for the remote transaction timeout values from the source bus 100 to the destination bus 120, the portal 141 on the intermediate bus 110 will intercept the TIMEOUT request message and synthesize the response.

This process significantly shortens the turnaround time for a TIMEOUT request by a node on the source bus 100, because the portal 141 on the intermediate bus retrieves the stored timeout values and calculates the total timeout values substantially faster than if a TIMEOUT request and TIMEOUT response were exchanged between nodes of the source bus 100 and destination bus 120.

This method also frees up the serial bus resources for servicing other nodes and for reducing overall transaction time of the network.

FIG 3 provides an overview of the steps of the above method according to one aspect of the present invention in terms of intercepting a TIMEOUT response message and storing its remote timeout values into internal register table.

At step 310, a TIMEOUT response message addressed to the portal's local bus is intercepted.

At step 320, the remote timeout values in the TIMEOUT response message intercepted in step 310 is stored in the register table corresponding to the destination bus ID.

At step 330, the intercepted TIMEOUT response message is forwarded to the originally addressed node.

Accordingly, steps 310-330 permit the storage of the remote timeout values for retrieval in subsequent requests, in order to enhance the efficiency of the protocol of the serial bus.

FIG 4 provides an explanation of the procedure for intercepting a TIMEOUT request message and synthesizing a corresponding TIMEOUT response message with remote timeout values.

At step 410, if a TIMEOUT request message whose remote transaction timeout values from the portal's local bus to the destination bus, to which the TIMEOUT request message is addressed, have been stored previously in its register table is received, step 415 will be processed next, otherwise the process returns to the start and step 410 will be the next step to execute.

At step 415, the received TIMEOUT request message is intercepted by the portal. Step 420 will be executed next.

At step 420, it is determined whether the source bus ID of the intercepted TIMEOUT request message is equal to the portal's local bus ID. If step 420 is answered affirmatively, step 425(I) is executed next. Otherwise, if step 420 is answered negatively, step 425(ii) is executed next. These two steps cover either scenario where the requester could be a node from the local bus, or from a remote source bus.

At step 425(I) the corresponding TIMEOUT response with the remote timeout values for the portal's local bus to the destination bus retrieved from the register table entry corresponding to the destination bus.

Alternatively, when step 425(ii) is performed, the corresponding TIMEOUT response message is synthesized, where its timeout values are calculated by the following procedure: The remote_timeout_seconds, the remote_ timeout_cycles and the hop_count values in the intercepted TIMEOUT request message are added to those in the table register entry corresponding to the destination bus ID, respectively. The max_remote_ payload value is set to a smaller value between that in the intercepted TIMEOUT request message and that stored in the table register entry corresponding to the destination bus ID.

At step 430 the synthesized TIMEOUT response message is sent to the original requester identified by the source ID of the intercepted TIMEOUT request message.

According to another aspect of the present invention, all bridges along the way between the source bus and the destination bus store the timeout value to the destination bus from its local bus in one shot. For example, if there are ten bridges between the source bus and the destination bus, up to ten bridges can store the appropriate timeout value for a single TIMEOUT message for future use. This greatly improves the efficiency of managing these TIMEOUT messages.

An exemplary embodiment of another method for communicating between nodes of a network according to another aspect of the present invention proceeds as follows. For clarity, we shall describe this method in terms of entry portal 131 in bridge 130, source bus 100 and destination bus 120; however, this method could be applied to any other portal, source bus or destination bus as well. Moreover, one or more entry portals in a bridge in a network may perform this method.

(A1) When a bridge (i.e., either an entry portal or an exit portal) 130 receives a TIMEOUT request message for a given destination bus, such as bus 120, the bridge check's its storage (e.g., entry portal 131 checks entry portal's storage 133) to determine whether the set of remote timeout values (e.g, T_x = remote timeout seconds, P_x = maximum remote payload, C_x = hop count) from the bridge's local bus (e.g., entry portal 131's local bus (i.e., bus 110)) to the destination bus 120 identified in the TIMEOUT request message is already known. This information can be stored in storage that is accessible to the bridge, as described with reference to FIG 1, in the form of flag indicating whether the associated stored set of remote timeout values is valid or not, e.g., a valid_flag for the destination bus identified in the TIMEOUT request message. A registry table similar to that in FIG 2 is included in each portal and stores on a per-bus basis, these remote timeout values along with the valid-flag. For example, as shown in FIG 2, the reserved field could be used to store the value of the valid flag.

In addition to having been previously determined, as set forth below, there are other ways this set of remote timeout values could be known to a given portal. For example, if the entry portal's co-portal is the exit portal on the destination bus, the entry portal knows the remote timeout seconds, the maximum remote payload, and the hop count values for the destination bus. Such is the case for entry portal 141 whose co-portal 142 (i.e., co-portals 141, 142 exist on the same bridge 140) is the exit portal (i.e., an exit portal is a portal coupled to the destination bus) for destination bus 120. Thus, entry portal 141 knows the remote timeout seconds value, the maximum remote payload value and the hop count value for destination bus 120.

In addition, the remote timeout seconds value can be calculated from known values, e.g., $\text{remote\_timeout} = \text{split\_timeout}_{\text{co-portal}} + \text{max\_forwarding\_time}_{\text{bridge}}$

in which:

remote_timeout is the value for the remote timeout seconds to be used in the timeout response message;

$\text{split\_timeout}_{\text{co-portal}}$ is the value included in the split timeout register of the entry portal's co-portal; and

$\text{max\_forwarding\_time}_{\text{bridge}}$ is the value known to the entry portal for the maximum forwarding time from the bridge to the destination bus.

The split_timeout register value of entry portal 131's co-portal (i.e., portal 132) is made available to entry portal 131 in a known manner, e.g., during initialization of the network or bridge 130.

Similarly, the internal maximum forwarding time (for both request and response subactions) for bridge 130 is also made available to entry portal 131. As such, entry portal 131 can then calculate the remote timeout seconds value for destination bus 120 as set forth above.

Similarly, the maximum remote payload value is also made available to entry portal 141 in a similar manner. The maximum remote payload value is a maximum payload size that can pass through the bridge, e.g., bridge 140. The hop count value is always one when the entry portal's (141) co-portal (142) is the exit portal (142) for the destination bus (bus 120). This set of values can be pre-stored by the entry portal together with the valid_flag for the destination bus set to "one" (1), thereby indicating that this set of values are valid for the designated destination bus. Thus, the register table could include the above entries for multiple destination buses, some of which are valid and some of which if stored are not yet valid.

(A2) If the set of remote timeout values is already known (e.g., the valid_flag is set equal to "1"), the bridge 130 (e.g., entry portal 131) synthesizes a TIMEOUT response message by determining the set of remote timeout values from known values or by using the exact stored values. The TIMEOUT response message is then sent back towards the source bus.

The remote timeout seconds value is determined by adding the known remote timeout seconds value ($T_{x,N,M}$) from the portal on the bridge's local bus (e.g., entry portal 131's local bus (bus 100)) to the destination bus 120 to the remote timeout seconds value ($T_{x,N}$) in the TIMEOUT request message. In other words,

$$T_{x,N\ response} = T_{x,\ N\ request} + T_{x,N,M}$$

in which:

$T_{x,N,M}$ is the value of remote timeout seconds from the Mth portal to the Nth bus (in this example portal 131 to bus 120), which is known to the Mth portal;

$T_{x,N\ response}$ is the value of the remote timeout seconds for the Nth bus sent in the timeout response message; and

$T_{x,N\ request}$ is the value of the remote timeout seconds for the Nth bus received in the timeout request message.

The bridge (e.g., entry portal 131) also compares the known maximum remote payload value ($P_{x,\ M}$) for the destination bus 120 from the local bus (e.g., entry portal 131's local bus (100)) with the maximum remote payload value ($P_{x,\ request}$) in the request message, and employs a smaller of the two values as the maximum remote payload value ($P_{x,\ response}$) in the response message. In other words,

$$P_{x\ response} = \min\left(P_{x\ request}, P_{x,\ M}\right)$$

in which:

$P_{x,\ response}$ is the value for the maximum remote payload included in the timeout response message;

$P_{x,\ request}$ is the value for the maximum remote payload included in the received timeout request message; and

$P_{x,M}$ is the value known to the entry portal for the maximum remote payload from the entry portal's local bus to the destination bus.

The bridge 130 (e.g., entry portal 131) also adds a known hop count value (C_M) to the destination bus 120 from the local bus (e.g., entry portal 131's local bus 100) to a hop count value ($C_{x\ request}$) in the request message and employs the sum as the hop count value in the timeout response message. In other words,

$$C_{x,\ response} = C_{x,\ request} + C_{x,M}$$

in which:

$C_{x,\text{response}}$ is the value for the hop count included in the timeout response message;

$C_{x,\text{request}}$ is the value for the hop count included in the received timeout request message; and

$C_{=x,M}$ is the value known to the entry portal for the hop count from the entry portal's local bus to the destination bus.

Otherwise, if the set of remote timeout values are not known (e.g., the valid_flag is not set to "1", or the valid_flag is set to "0"), the bridge 130 (e.g., entry portal 131) will perform the procedure set forth in steps (A3a) and (A3b) below.

(A3a) First, bridge 130 temporarily stores in the table of FIG 2, for example, the received set of remote timeout values (e.g., T_0, P_0, and C_0) for destination bus 120 with the valid_flag cleared, in which T_0 is the remote_timeout seconds value, P_0 is the max_remote_payload value, and C_0 is the hop_count value. This set of remote timeout values are all included in the received TIMEOUT request message.

(A3b) Next, the bridge 130 calculates a set of new remote timeout values. This set of calculated timeout values is calculated as follows.

The temporarily stored remote timeout seconds value (T_0) is increased by a calculated value, $T_{x,M}$. In other words:

$$T_{x\,request} = T_0 + T_{x,M}$$

in which

$$T_{x,M} = T_{x,M\,request} + T_{x,\,M\,response}\,.$$

Thus, the calculated value, $T_{x,\,M}$, is a sum of a maximum forward time for request subactions $T_{x,\,M\,\text{request}}$ (from entry portal 131 to an entry portal 141 of a next bridge 140) and a maximum forward time for response subactions $T_{x,\,M\,\text{response}}$ (from entry portal 131's co-portal (i.e., portal 132) (i.e., the entry portal for response subactions) to either an entry portal 141 of a next bridge 140 (from the perspective of response subactions) or the requester node (e.g., node 151).

The maximum payload value ($P_{x\,\text{request}}$) is set as P_x,

$$P_{x\,request} = P_x$$

in which P_x is a maximum data payload that can be forwarded from entry portal 131 to the entry portal 141 of the next bridge 140.

The temporarily stored hop count value (C_0) is increased by one. In other words,

$$C_{x\,request} = C_0 + 1$$

in which $C_{x\ request}$ is the output hop count value in the output timeout request message.

The bridge 130 then forwards the newly calculated set of remote timeout values (e.g., $T_{x\ request}$, $P_{x\ request}$, $C_{x\ request}$) toward the destination bus 120 as part of a TIMEOUT request message destined for destination bus 120.

(A3c) Subsequently, the bridge 130 (e.g., entry portal 131) intercepts a corresponding TIMEOUT response message that includes a set of timeout response values (e.g., T_y, P_y, C_y). This corresponding TIMEOUT response message is a TIMEOUT response message in reply to the TIMEOUT request message sent in step (A3b)). The bridge (e.g., entry portal 131) then calculates a new set of remote timeout values.

For the remote timeout seconds value to be used in future timeout responses messages for destination bus 120, $T_{x,\ M\ response}$, the bridge 130 (e.g., entry portal 131) calculates:

$$T_{x,M\ response} = T_y - T_0$$

in which T_y is the remote timeout seconds value received in the intercepted timeout response message, and T_0 is the previously stored value of the timeout response seconds received in step (A3a).

For the maximum remote payload value to be used in future timeout response messages for destination bus 120, $P_{x,\ M\ response}$, the bridge (e.g., entry portal 131) uses the received maximum payload value, P_y, that is:

$$P_{x,M\ response} = P_y$$

For the hop count value to be used in future timeout response messages for destination bus 120, $C_{x,\ M\ response}$, the bridge (e.g, entry portal 131) calculates the difference between the previously stored hop count value (C_0) and the received hop count value, C_y, which is:

$$C_{x,M\ response} = C_y - C_0$$

The entry portal then stores the newly calculated set of remote timeout values (i.e., $T_{x,\ M\ response}$, $P_{x,\ M\ response}$, $C_{x,\ M\ response}$) for future use with the valid flag set, and forwards a new set of remote timeout values (T_y, $\min(P_y, P_0)$, C_y) towards the source bus as part of a timeout response message, in which T_y is the remote timeout seconds value, $min(P_y, P_0)$ is the maximum remote payload value, and C_y is the hop count value.

According to another aspect of the present invention, another exemplary embodiment for producing the same result as the immediately preceding embodiment operates as follows.

(B1) This step operates as step (A1) above. Thus, the bridge (e.g., entry portal) checks its storage to determine if the set of remote timeout values is known, which will occur as described above in (A1).

(B2) This step also operates as step (A2) above. Thus, if the set of remote timeout values are known, then the bridge (e.g., entry portal) synthesizes the timeout response message as set forth in step (A2).

Otherwise (e.g., the valid_flag is not set or equals 0), the bridge (e.g., entry portal) will perform the following procedure:

(B3a) This step also operates as step (A3) above. Thus, the bridge (e.g., entry portal) temporarily stores the received set of remote timeout values for the destination bus with the valid flag cleared.

(B3b) Next, the bridge (e.g., entry portal) calculates a set of new remote timeout values. This set of calculated timeout values is calculated as follows.

The temporarily stored remote timeout seconds value (T_0) is increased by the calculated value, $T_{x,M}$, as described above In other words:

$$T_{x\ request} = T_0 + T_{x,M}$$

in which

$$T_{x,M} = T_{x,M\ request} + T_{x,\ M\ response} \, .$$

The maximum payload value ($P_{x\ request}$) is set to 0xFFFF, where 0xFFFF is the maximum value for the maximum remote payload field, i.e.,

$$P_{x\ request} = 0xFFFF \, .$$

The temporarily stored hop count value (C_0) is increased by one. In other words,

$$C_{x\ request} = C_0 + 1$$

in which $C_{x\ request}$ is the output hop count value in the output timeout request message.

The bridge (e.g., entry portal) then forwards the newly calculated set of remote timeout values (e.g., $T_{x\ request}$, $P_{x\ request}$, $C_{x\ request}$) toward the destination bus 120 as part of a TIMEOUT request message destined for destination bus 120.

(B3c) Subsequently, the bridge (e.g., entry or exit portal) intercepts a corresponding TIMEOUT response message that includes a set of timeout response values (e.g., T_y, P_y, C_y). This corresponding TIMEOUT response message is a TIMEOUT response message in reply to the TIMEOUT request message sent in step (B3b)). The bridge (e.g., entry portal) then calculates a new set of remote timeout values.

For the remote timeout seconds value to be used in future timeout response messages for destination bus 120, $T_{x,\ M\ response}$, the bridge (e.g., entry portal) calculates:

$$T_{x,M\ response} = T_y - T_0$$

in which T_y is the remote timeout seconds value received in the intercepted timeout response message, and T_0 is the previously stored value of the timeout response seconds received in step (B3a).

For the maximum remote payload value to be used in future timeout response messages for destination bus 120, $P_{x,\ M\ response}$, the bridge (e.g., entry portal) calculates the minimum of the received maximum payload value, P_y, and the maximum data payload (P_x) that can be forwarded from entry portal 131 to the entry portal 141 of the next bridge 140, that is:

$$P_{x,M\ response} = \min(P_y, P_x).$$

For the hop count value to be used in future timeout response messages for destination bus 120, $C_{x,\ M\ response}$, the bridge (e.g., entry portal) calculates the difference between the previously stored hop count value (C_0) and the received hop count value, C_y, which is:

$$C_{x,M\ response} = C_y - C_0$$

The bridge (e.g., entry portal) then stores the newly calculated set of remote timeout values (i.e., $T_{x,\ M\ response}$, $P_{x,\ M\ response}$, $C_{x,\ M\ response}$) for future use with the valid flag set, and forwards a new set of remote timeout values (T_y, min(P_y, P_x, P_0), C_y) towards the source bus as part of a timeout response message, in which T_y is the remote timeout seconds value, min(P_y, P_x, P_0) is the maximum remote payload value, and C_y is the hop count value.

The above-discussed procedures provide significant efficient improvements over the earlier described technique without requiring changes to the draft standard. This technique works even if there are both legacy bridges (i.e., bridges based on the draft standard) and these enhanced bridges mixed on the network.

However, further improvement and simplification are possible if changes to the draft standard are allowed and all the bridges follow the same rules as described below.

(C1) As with the immediately above two exemplary embodiments, this step operates as steps (A1) and (B1). However, in this case, each bridge (e.g., entry portal) checks the destination of a received TIMEOUT request message to determine whether the set of remote timeout values is known for the identified destination bus by checking the status of the valid flag in memory.

(C2) If the set of remote timeout values is known (e.g., valid_flag=1), each bridge (e.g., entry portal) generates a corresponding TIMEOUT response message by inserting the stored remote_timeout, max_remote_payload, and hop_count values in the proper fields in the TIMEOUT response message in the manner as described above.

(C3a) If the set of remote timeout values is not known to each bridge, (e.g., valid_flag=0), each bridge forwards the TIMEOUT request message towards the destination without modifying the set of remote timeout values in the TIMEOUT request message. The value of each field in the message shall be (0, 0xFFF, 0) as initialized by the requester by definition.

(C3b) If a bridge (from the perspective of response subactions) receives a TIMEOUT response message, the bridge (e.g., exit portal) 132 calculates a set of new remote timeout values for storage and use in subsequent TIMEOUT response messages with the valid flag set. This set of calculated timeout values is calculated as follows.

The received remote timeout seconds value (T_y) is increased by the calculated value, $T_{x,M}$, as described above In other words:

$$T_{x\ response} = T_y + T_{x,M}$$

in which

$$T_{x,M} = T_{x,M\ request} + T_{x,\ M\ response}\ .$$

The smaller of the received maximum remote payload value (P_y) and the maximum data payload (P_x) that can be forwarded from exit portal 132 to the entry portal 141 of the next bridge 140 is determined, that is:

$$P_{x,M\ response} = \min(P_y, P_x).$$

The received hop count value (C_y) is increased by one. In other words,

$$C_{x\ response} = C_y + 1$$

in which $C_{x\ response}$ is the output hop count value in the output timeout response message.

These calculated values ($T_{x,\ response}$, $P_{x,\ response}$, $C_{x,\ response}$) are then stored for future use in timeout response messages with the valid flag set (e.g., valid_flag=1). The same values are then forwarded as part of a TIMEOUT response message.

For all of the above-mentioned procedures, a table with entries each consisting of remote_timeout seconds field, max_remote_payload field, hop_count field, and valid_flag can be used to store both temporary and final remote timeout parameters for each destination bus. The table entries shall be cleared upon net-topology change.

Turning to FIG 5, shown therein is a flowchart of an exemplary embodiment 50 of the above-described methods for efficiently communicating timeout messages in a bridged network.

In step 51, the portal checks the valid flag in the register table upon receipt of a timeout request message to determine if the set of stored remote timeout values is valid.

In step 52, if the valid_flag is one, the process moves to step 53, otherwise the process moves to step 54.

In step 53, the portal synthesizes a timeout response message using the stored set of remote timeout values, as set forth in (A1) and (A2) above.

In step 54, the received set of remote timeout values is temporarily stored with the valid flag cleared.

In step 55, a first new set of remote timeout values is determined and forwarded as part of the timeout request message towards the destination bus. This first new set of remote timeout values can be determined as set forth in (A3b) or (B3b) above.

In step 56, a timeout response message is intercepted by the portal, which timeout response message includes a set of remote timeout response values.

In step 57, a second new set of remote timeout values is determined and stored for later use with the valid flag set based on the received set of remote timeout response values. This second new set of remote timeout values can be determined as set forth in (A3c) or (B3c) above.

In step 58, a third new set of remote timeout values is determined and forwarded as part of a timeout response message to the source bus based on the received set of remote timeout response values. This third new set of remote timeout values can be determined as set forth in (A3c) or (B3c) above.

Turning to FIG 6, shown therein is a flowchart of an exemplary embodiment 60 of the above-described methods for efficiently communicating timeout messages in a bridged network.

In step 61, each portal checks the valid flag in the register table upon receipt of a timeout request message to determine if the set of stored remote timeout values is valid.

In step 62, if the valid_flag is one, the process moves to step 63, otherwise the process moves to step 64.

In step 63, the portal synthesizes a timeout response message using the stored set of remote timeout values, as set forth in (C1) above.

In step 64, the timeout request message is forwarded towards the destination bus without modifying the set of remote timeout values that was included in the received timeout request message.

In step 65, a portal receives a timeout response message that includes a set of remote timeout response values.

In step 66, the stored set of remote timeout values is updated using the received set of remote timeout response values. The stored set of remote timeout values is updated as set forth in (C3b) above.

In step 67, the valid flag is set.

In step 68, the updated set of remote timeout values is forwarded towards the source bus as part of a timeout response message.

Although various embodiments are specifically illustrated and described herein, it will be appreciated that modifications and variations of the invention are covered by the above teachings and are within the purview of the appended claims without departing from the spirit and intended scope of the invention. For example, names for certain parameters are used, such as max_remote_payload, and hop_count, however, other names could be employed as well. Moreover, the terms bridge, entry portal, exit portal and portal are used to describe various devices that perform the messaging functions herein. Each of these devices could perform the messaging functions and calculations set forth herein. Furthermore, these examples should not be interpreted to limit the modifications and variations of the invention covered by the claims but are merely illustrative of possible variations.